SOFTWARE DESIGN SPECIFICATION

FOR A ONE RUNWAY AIRPORT/AIR TRAFFIC CONTROLLER SIMULATION

BY: PATRICIA BURKE, KENNETH MARTIN, DIANE LONGTIN

VERSION: REVISION – 1.0

OCTOBER 19, 2001

REVISION CHART

.

Version	Primary Author(s)	Description of Version	Date Completed
Draft	Tricia Burke Kenneth Martin Diane Longtin	Initial draft created for distribution and review comments	10/15/2001
Revision – 1.0	Tricia Burke Kenneth Martin Diane Longtin	First Release	10/22/2001
Revision – 1.1	Tricia Burke Kenneth Martin Diane Longtin	Final Release	10/24/01

Contents

1. INTRODUCTION 6

1.1 Purpose 6

1.2 Scope 6

1.3 Objective 6

2. SYSTEM OVERVIEW 7

2.1 Product Perspective 7
2.1.1 Design Method 7
2.1.1 User Interfaces 7
2.1.2 Hardware Interfaces 7
2.1.3 Software Interfaces 7
2.1.4 Memory Constraints 7
2.1.5 Operations 7
2.1.6 Site Adaptation Requirements 7

2.2 Product Functions 8

2.3 User Characteristics 8

2.4 Constraints 8

2.5 Assumptions and Dependencies 8

2.6 Apportioning of Requirements 8

3. DESIGN CONSIDERATIONS 9

3.1 Operating Environment 9

3.2 Fault Tolerant Design 9

3.3 Design Conventions 9

3.4 Architectural Design 9

3.5 User Interface: 9
3.5.1 Expected Input: 9

3.5.2 Output: 10

4. SYSTEM ARCHITECTURE 111

4.1 View of Product Classes (Figure 1) 111

4.2 Individual Classes of System 122
4.2.1 Airport: 12
4.2.1.1 Attributes of Airport class 122
4.2.1.1.1 takeoffQueue 122
4.2.1.1.2 landingQueue 122
4.2.1.2 Functions available in Airport Class 122
4.2.1.2.1 Averager ( ) 122
4.2.1.2.2 newArrival ( ) 122
4.2.2 Airplane 122
4.2.2.1 Functions available in Airplane Class 12
4.2.2.2.1 takeoff ( ) 12
4.2.2.2.2 land ( ) 133
4.2.2.2.3 crash ( ) 133
4.2.3 Runway 133
4.2.3.1 Functions available in Runway Class 133
4.2.3.1.1 isBusy ( ) 133
4.2.3.1.2 moveToRunway ( ) 133
4.2.3.1.3 timeleftonRunway ( ) 133
4.2.4 Queue 133
4.2.4.1 Functions available in Queue class 133
4.2.4.1.1 size ( ) 133
4.2.4.1.2 Capacity ( ) 133
4.2.4.1.3 insert ( ) 144
4.2.4.1.4 front ( ) 144
4.2.5 Averager 144
4.2.5.1 Attributes of Averager Class 144
4.2.5.1.1 count 144
4.2.5.1.2 sum 144
4.2.5.1 Functions available in Averager Class 144
4.2.5.2.1 next_number ( ) 144
4.2.5.2.2 average ( ) 144
4.2.6 NextArrival 144
4.2.6.1 Attributes of NextArrival Class 144
4.2.6.1.1 aver_wait_time 144
4.2.6.2 Functions available in NextArrival Class 144
4.2.6.2.1 wait_time ( ) 144

5. FIGURES **155**

5.1 Use Cases (Figure 2) **155**

5.2 Successful Takeoff **166**
5.2.1 Successful Takeoff Sequence (Figure 3) 166
5.2.2 Successful Takeoff Collaboration (Figure 4) 177

5.3 Successful Landing **188**
5.3.1 Successful Landing Sequence (Figure 5) 188
5.3.2 Successful Landing Collaboration (Figure 6) 199

5.4 Runway Busy **20**
5.4.1 Runway Busy Sequence (Figure 7) 20
5.4.2 Runway Busy Collaboration (Figure 8) 211

5.5 Crash **222**
5.5.1 Crash Sequence (Figure 9) 222
5.5.2 Crash Collaboration (Figure 10) 233

5.6 Print Report **244**
5.6.1 Print Report Sequence (Figure 11) 244
5.6.3 Print Report Collaboration (Figure 12) 255

6. REFERENCES **266**

6.1 References **266**

1. INTRODUCTION

1.1 Purpose

This document will define the design of the one runway simulator. It contains specific information about the expected input, output, classes, and functions. The interaction between the classes to meet the desired requirements are outlined in detailed figures at the end of the document.

.

1.2 Scope

This Design Specification is to be used by Software Engineering and Software Quality Engineering as a definition of the design to be used to implement the One Runway Airport/Air Traffic Simulator

1.3 Objective

The One Runway Airport/Air Traffic Controller Simulation System is established to simulate a one runway airport that supports takeoff and landings of airplane. Its purpose is to establish the average time between runway activities that the one runway airport can support. It records the number of planes processed and their average time spent in waiting. It also records planes whose time spent in the queue exceeds the maximum fuel allotment

2. SYSTEM OVERVIEW

.

2.1 Product Perspective

This product simulates a one-runway airport that supports takeoff and landings of planes. It records the number of planes processed and their average time spent in waiting. It also records planes whose time spent in queue exceeded the maximum fuel allotment. Depending on the values that are used by the simulator operator, it can be determined what is the optimal way to utilize a one runway airport to achieve maximum takeoffs/landings without causing any crashes.

.

2.1.1 Design Method

The design of this product utilizes an object-oriented approach.

2.1.1 User Interfaces

The user of this software product will be interfacing with the simulation system to help predict the behavior an actual air traffic control system. The product allows the user to get familiar with the software without actually having the responsibility of controlling the air traffic

2.1.2 Hardware Interfaces

This simulation software can run on most SPARC stations.

2.1.3 Software Interfaces

This simulator will execute on a Solaris UNIX platform running GNU C++ compiler.

.

2.1.4 Memory Constraints

This program takes up about 7 kb of memory. The output reports are modest in size and take up about 7 kb.

2.1.5 Operations

The operator will be required to enter the parameters for the simulation run from an operator console.

2.1.6 Site Adaptation Requirements

This software is intended to execute on any UNIX platform with no modifications needed to support different sites.

2.2 Product Functions

Airplane Arriving-This function calculates the time of arrival and clears the airplane for landing.

Wait in Queue-This function calculates the priority and statuses of the airplanes accordingly.

Arrive on Runway-This function confirms arrival and departure. It calculates average time.

Airplane Departing-This function calculates the time of arrival and clears the airplane for departure.

2.3 User Characteristics

The general characteristics of the intended users, include

- *educational level –bachelor of science*
- *experience- air traffic control knowledge*

2.4 Constraints

This application can only run on a system that supports GNU C++ compiler.

2.5 Assumptions and Dependencies

This simulation is only intended to report on takeoff and landing patterns.

2.6 Apportioning of Requirements

There are requirements apportioned to later releases of the simulator.

3. Design Considerations

3.1 Operating Environment

The One Runway Airport/Air Traffic Controller Simulation is intended to be operated in a Solaris Unix environment.

3.2 Fault Tolerant Design

Application errors will be handled by common fault detection services (e.g. common C++ exception handling, and error checking on task processing).

3.3 Design Conventions

The OneRunway software design uses the Object Oriented methodology described in "The Unified Software Development Process" by Ivar Jacobsen, Grady Booch and James Rumbaugh. (Booch, 1999)

3.4 Architectural Design

The software capabilities and requirements specified in the One Runway Airport/Air Traffic Controller Simulation Software Requirements Specification are transformed into programs that will execute on a Solaris. Software items are partitioned into classes and packages using Object Oriented methodology to maximize encapsulation and minimize interfaces. Packages are then built (compiled and linked) into executable programs.

3.5 User Interface:

The user or simulator operator interfaces via a text input screen. The user is prompted for several values in order to perform the calculations.

3.5.1 Expected Input:

The user is prompted to enter values for the following variables:

Time in minutes to land
Time in minutes to takeoff
Average time between arrivals
Average time between departures
Maximum time in arrival queue (indicates how long before all fuel consumed)
Number of minutes in simulation

3.5.2 Output:

The user receives a report from the simulator indicating how many planes were able to depart and how many were able to arrive in the simulation time. They also receive data indicating the average time a plane waited to land/takeoff, as well as how many planes crashed, or ran out of fuel while waiting.

4. System Architecture

4.1 View of Product Classes (Figure 1)

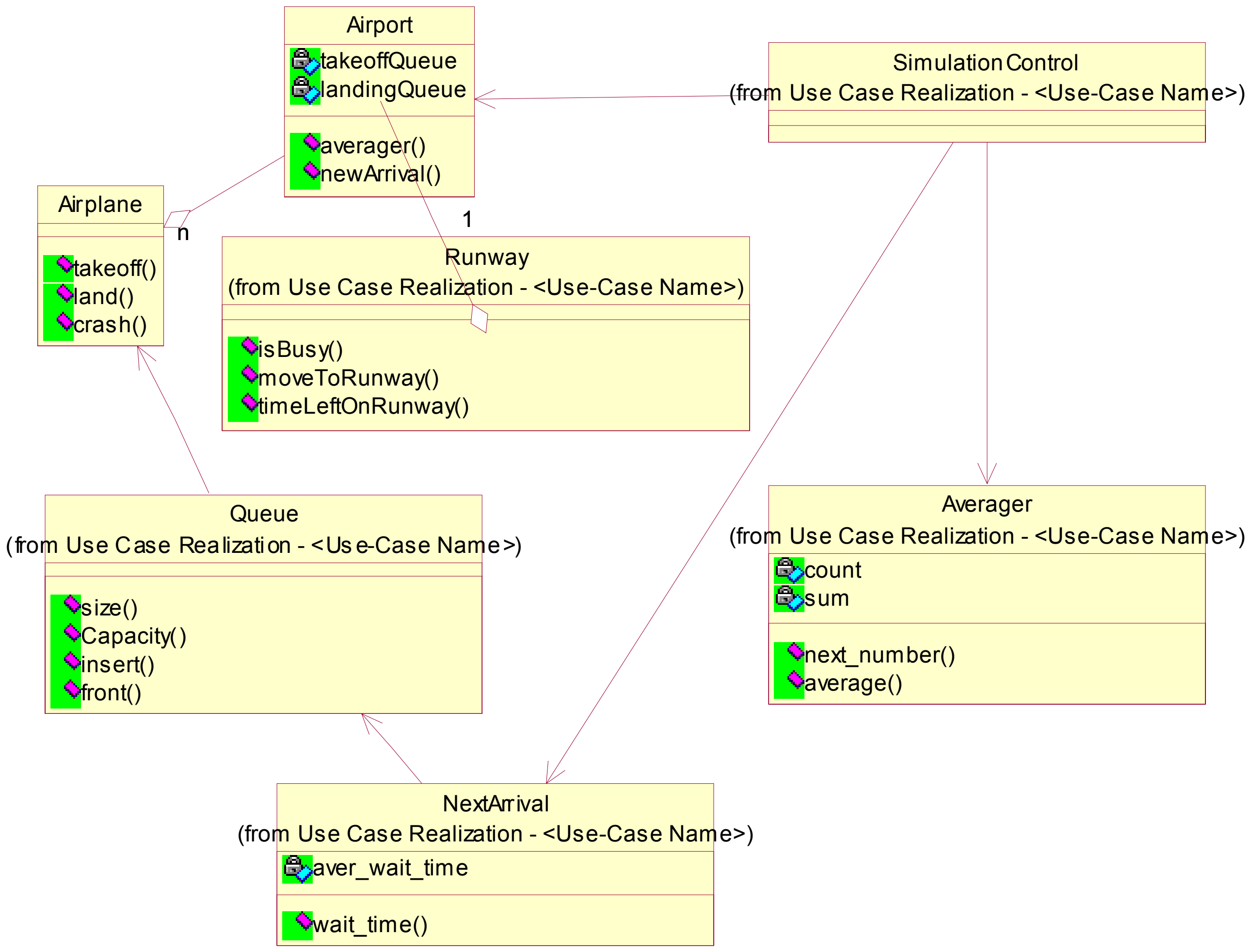


Figure 1- One Runway View of Participating Class Diagram

This figure shows the overall design of the system with outlining the individual classes and their relation to each other.

4.2 Individual Classes of System

4.2.1 Airport:

The airport class is where the data about the planes that are waiting to take off and land is kept. This class utilizes two queues as the data structure to hold the planes, one for takeoffs waiting and another for planes waiting to land.

4.2.1.1 Attributes of Airport class

4.2.1.1.1 TAKEOFFQUEUE

The takeoffQueue holds the data that for airplanes that are waiting to use the runway to takeoff.

4.2.1.1.2 LANDINGQUEUE

The landingQueue holds the data for the planes that are still in the air waiting to use the runway to land.

4.2.1.2 Functions available in Airport Class

4.2.1.2.1 AVERAGER ( )

Function: performs calculations on data that are used in the output report.
Precondition: All values needed for calculation are accessible to this function.
Postcondition: Necessary calculations are complete.

4.2.1.2.2 NEWARRIVAL ( )

Function: adds new plane to either takeoffQueue or landingQueue.
Precondition: takeoff and landing queues have been instantiated, and there is room on the queue for another plane.
Postcondition: New plane is on the queue.

4.2.2 Airplane

The airplane class simulates airplanes arriving at the airport and airplanes, which are ready for departure. A random number generator is utilized to randomly add planes for takeoff and add planes that are requesting to land.

4.2.2.1 Functions available in Airplane Class

4.2.2.2.1 TAKEOFF ( )

Function: Generate a plane requesting a takeoff.
Precondition: The average arrival time is greater than one minute.
Postcondition: The next plane requesting takeoff has been initialized and q (average time between takeoffs) is at least 1 minute.

4.2.2.2.2 LAND ( )
Function: Generate plane requesting to land.
Precondition: The average arrival time is greater than one minute.
Postcondition: The next plane requesting to land has been initialized and q (average time between landings) is at least 1 minute.

4.2.2.2.3 CRASH ( )
Function: Simulate a crash of plane due to it being in the queue too long and running out of fuel.
Precondition: Planes are in landing queue for greater than specified period.
Postcondition: Plane in landing queue has been removed from queue and number Plane crashing has been increased.

4.2.3 Runway

4.2.3.1 Functions available in Runway Class

4.2.3.1.1 ISBUSY ( )
Function: Indicates to calling function if runway is busy and can't accept a new takeoff or landing.
Precondition: Plane has requested to land or takeoff and isBusy is true.
Postcondition: Plane remains in queue. IsBusy remains true.

4.2.3.1.2 MOVETORUNWAY ( )
Function: Removes plane from queue and begins to reinitializes time counter that when takeoff or landing is complete.
Precondition: The runway is not busy.
Postcondition: The runway has begun to simulate one takeoff/landing cycle. At this point isBusy is set to true to indicate that no other planes are allowed on the runway.

4.2.3.1.3 TIMELEFTONRUNWAY ( )
Function: Keeps track of how much time is left of simulated takeoff/landing.
Precondition: isBusy is true indicating runway is in use.
Postcondition: Simulation of 1 minute has occurred and isBusy is set to false.

4.2.4 Queue

This class is utilized by the airport class to instantiate a takeoffQueue and landingQueue.

4.2.4.1 Functions available in Queue class

4.2.4.1.1 SIZE ( )
Function: Instantiates a queue of size n.
Precondition: Object of type queue class has been created.
Postcondition: New queue of size n has been created.

4.2.4.1.2 CAPACITY ( )
Function: Indicates number of items currently in queue.
Precondition: Item of queue class has been instantiated.

Postcondition: Numbers of items in queue returned.

4.2.4.1.3 INSERT ( )
Function: Adds new item at end of queue.
Precondition: queue is not full.
Postcondition: there is a new item at end of queue.

4.2.4.1.4 FRONT ( )
Function: Returns value of item at front of queue.
Precondition: Queue is not empty.
Postcondition: Value of front item has been returned.

4.2.5 Averager

4.2.5.1 Attributes of Averager Class

4.2.5.1.1 COUNT
Total number of items sent to averager.

4.2.5.1.2 SUM
Total sum of all items sent to averager.

4.2.5.1 Functions available in Averager Class

4.2.5.2.1 NEXT_NUMBER ( )
Function: Accepts next number to be summed for the average calculation.
Precondition: Averager is ready to accept numbers.
Postcondition: Next number has been added to sum and count is incremented by one.

4.2.5.2.2 AVERAGE ( )
Function: Uses sum and count to calculate average of all numbers.
Precondition: Count is greater than zero.
Postcondition: Average of summed numbers is returned.

4.2.6 NextArrival

4.2.6.1 Attributes of NextArrival Class

4.2.6.1.1 AVER_WAIT_TIME

4.2.6.2 Functions available in NextArrival Class

4.2.6.2.1 WAIT_TIME ( )
Function: Indicates how long until next arrival.
Postcondition: Amount of time until next arrival is returned.

5. FIGURES

5.1 Use Cases (Figure 2)

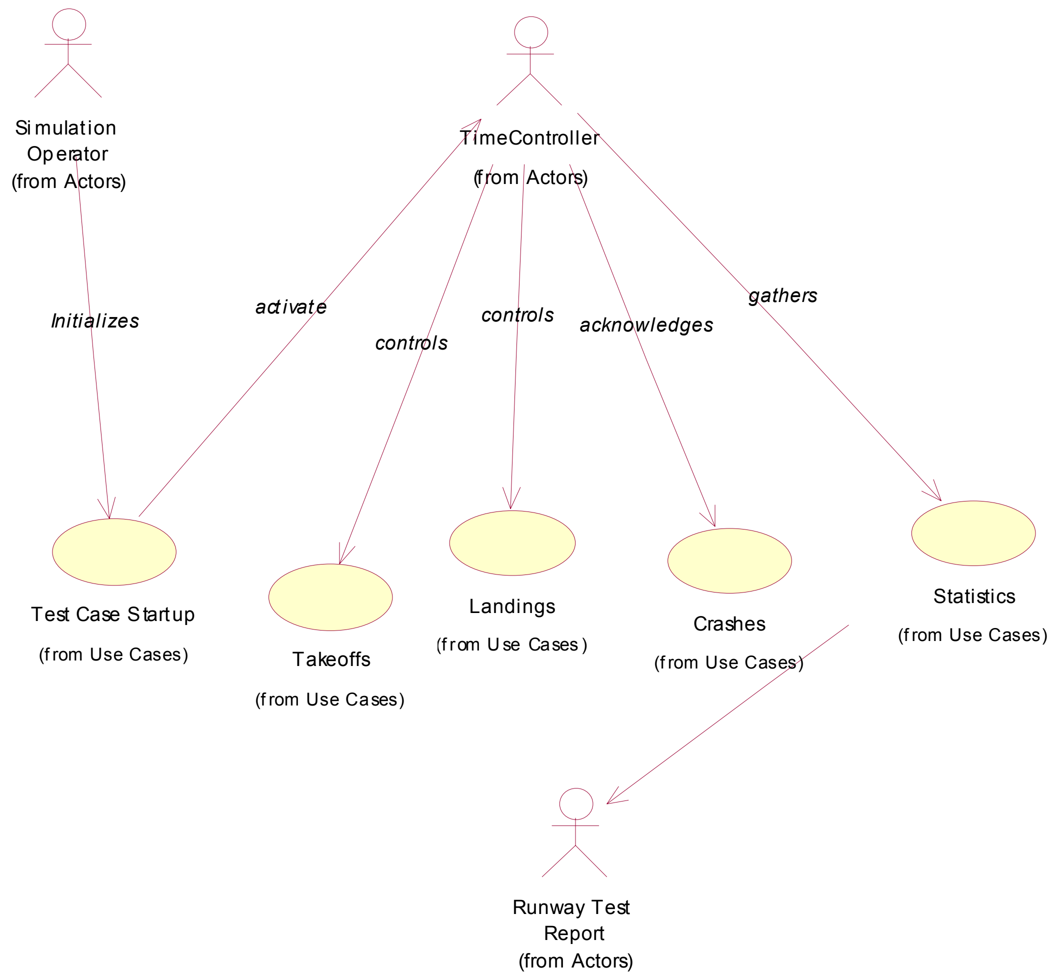

Simulation Operator (from Actors)
TimeController (from Actors)
Initializes
activate
controls
controls
acknowledges
gathers
Test Case Startup (from Use Cases)
Takeoffs (from Use Cases)
Landings (from Use Cases)
Crashes (from Use Cases)
Statistics (from Use Cases)
Runway Test Report (from Actors)

Figure 2- One Runway Use Cases

5.2 Successful Takeoff

5.2.1 Successful Takeoff Sequence (Figure 3)

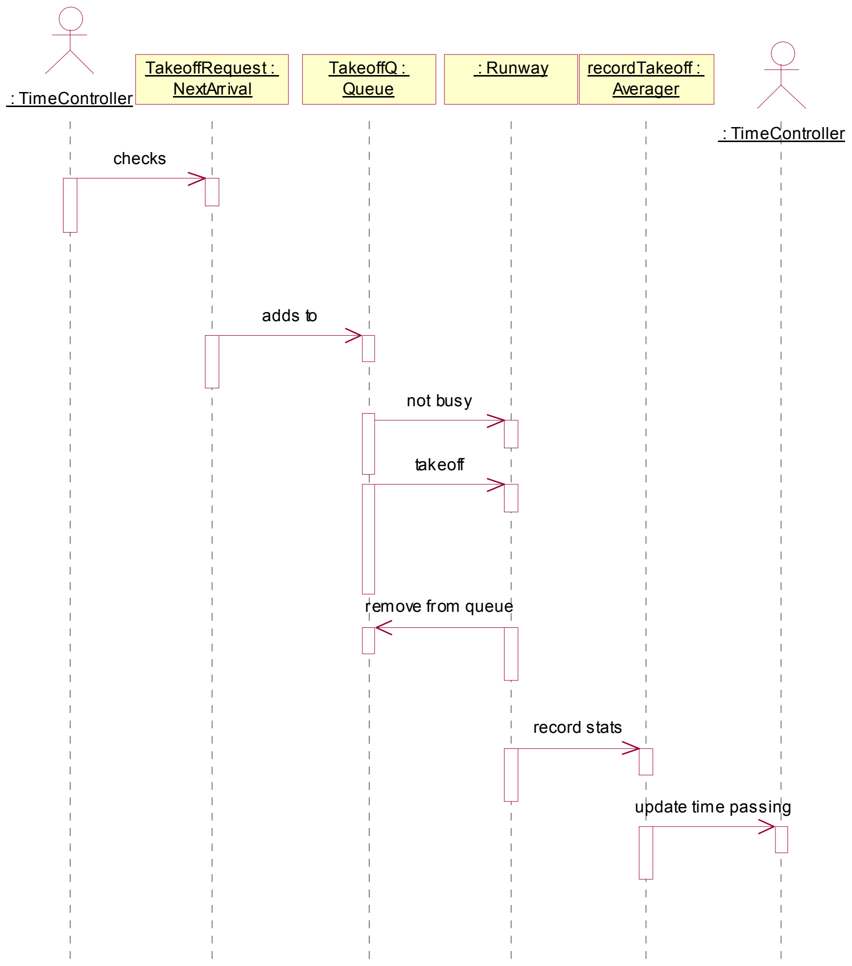


Figure 3- Successful Takeoff Sequence

5.2.2 Successful Takeoff Collaboration (Figure 4)

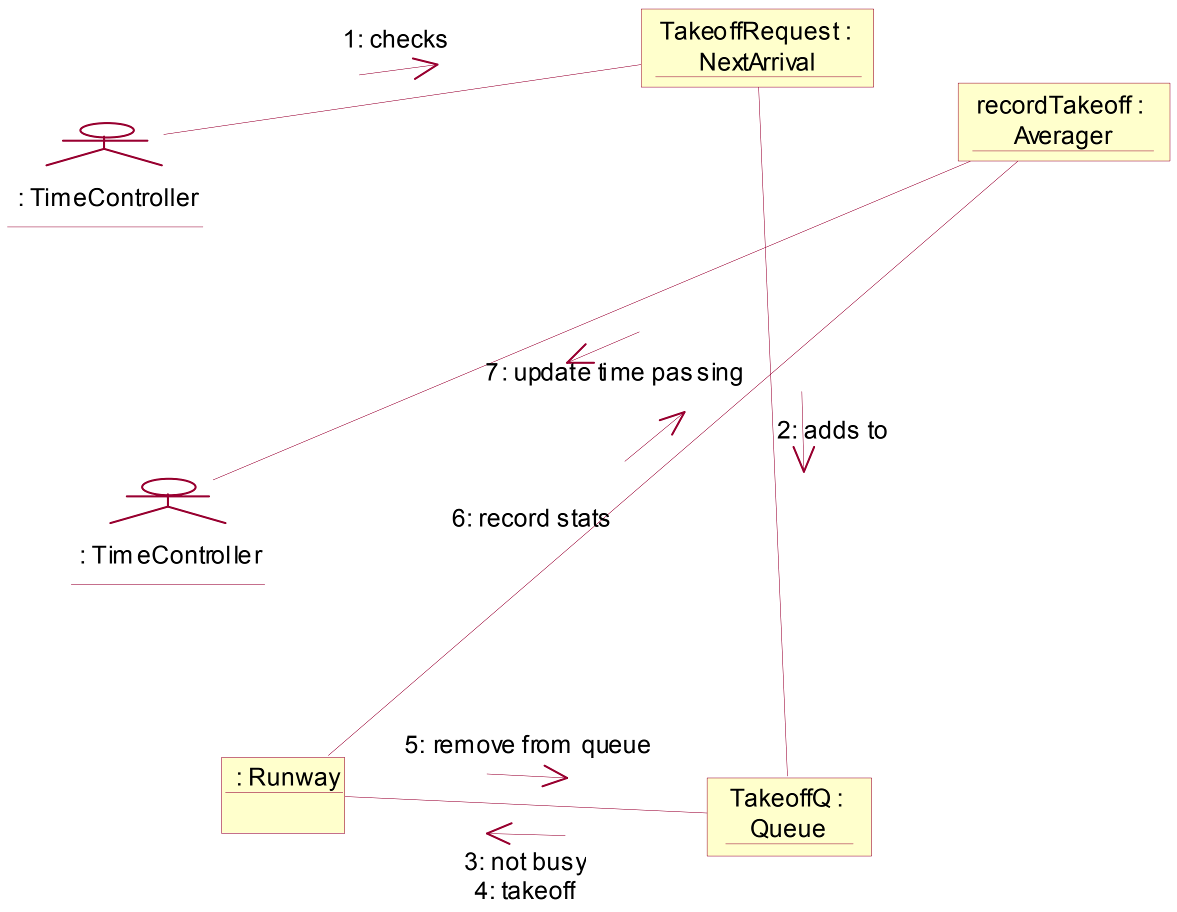


Figure 4- Successful Takeoff Collaboration

5.3 Successful Landing

5.3.1 Successful Landing Sequence (Figure 5)

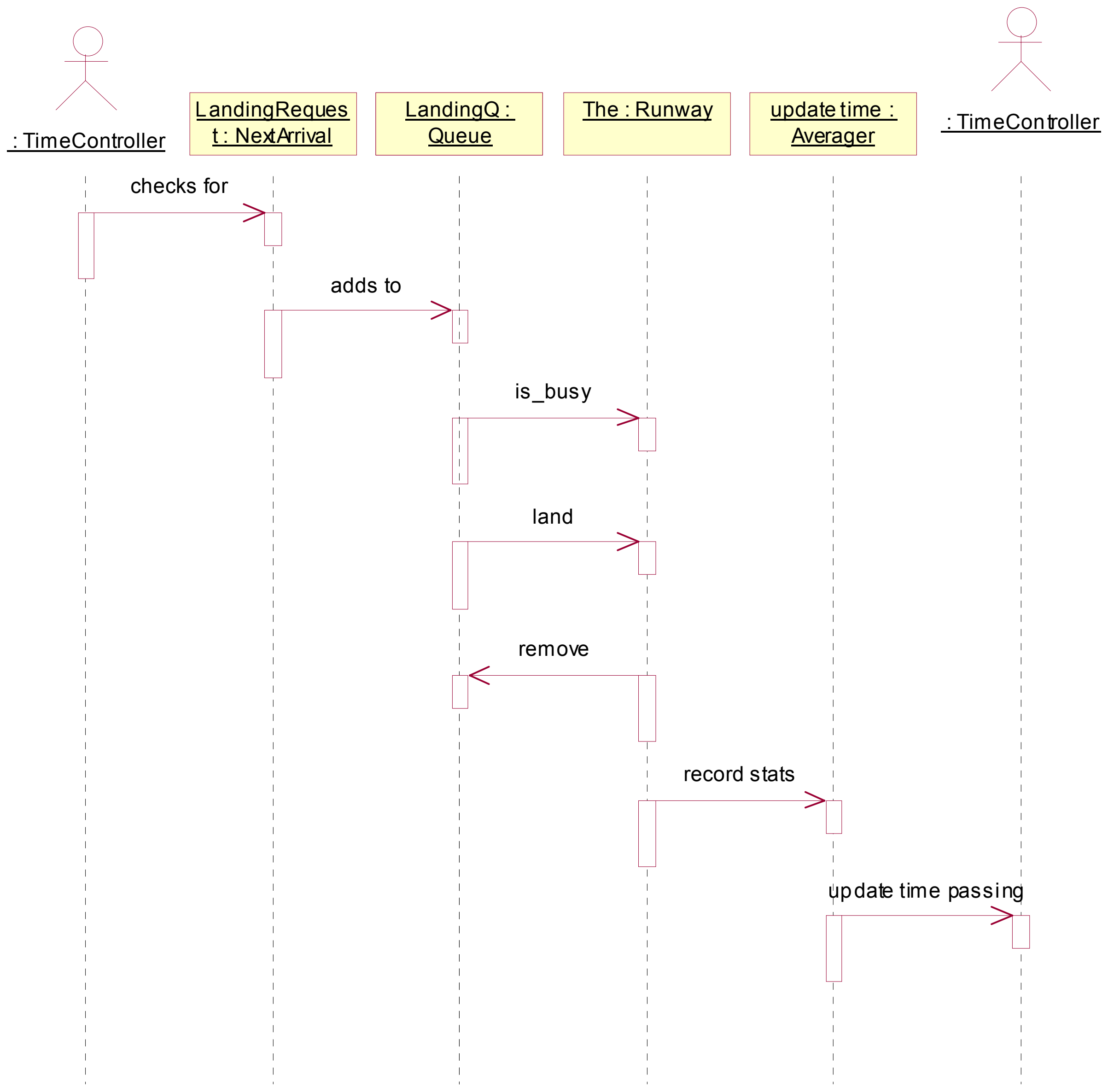


Figure 5- Successful Landing Sequence

5.3.2 Successful Landing Collaboration (Figure 6)

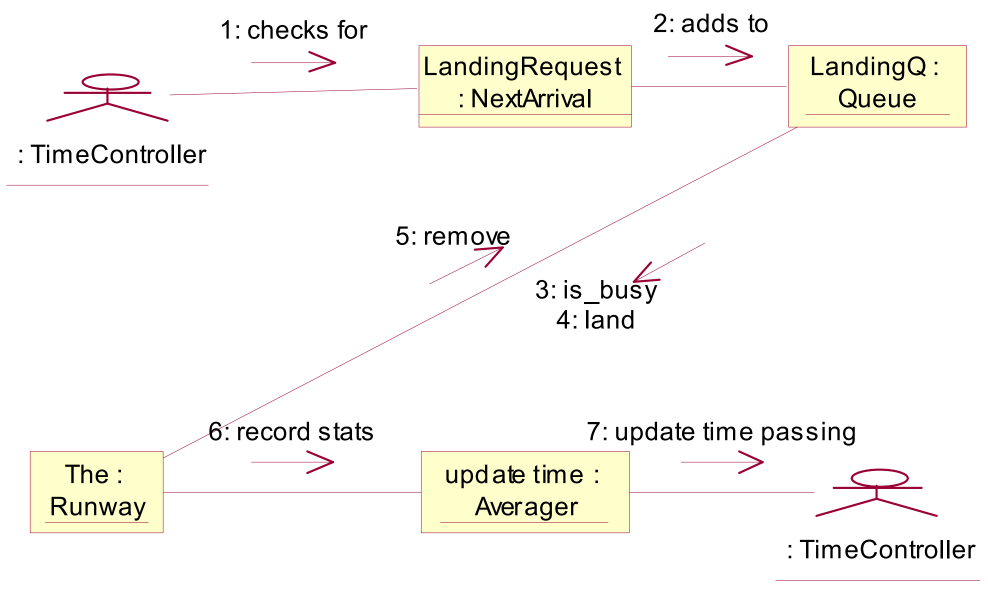


Figure 6 - Successful Landing Collaboration

5.4 Runway Busy

5.4.1 Runway Busy Sequence (Figure 7)

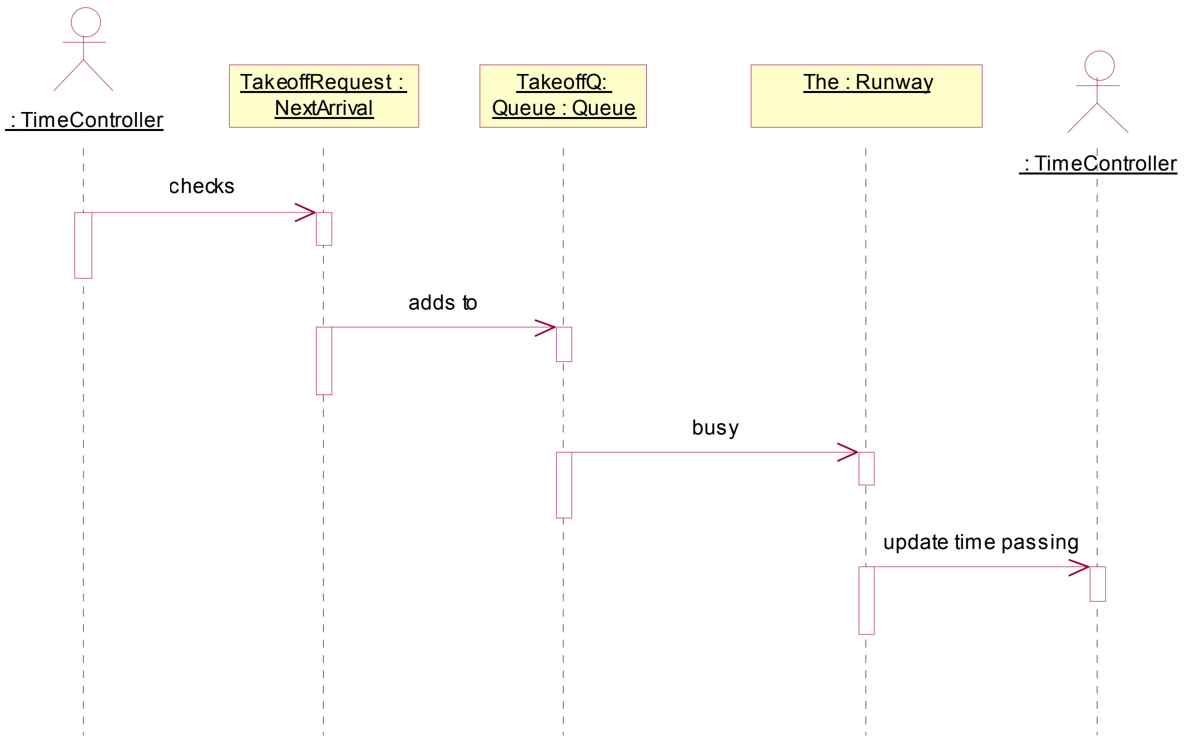
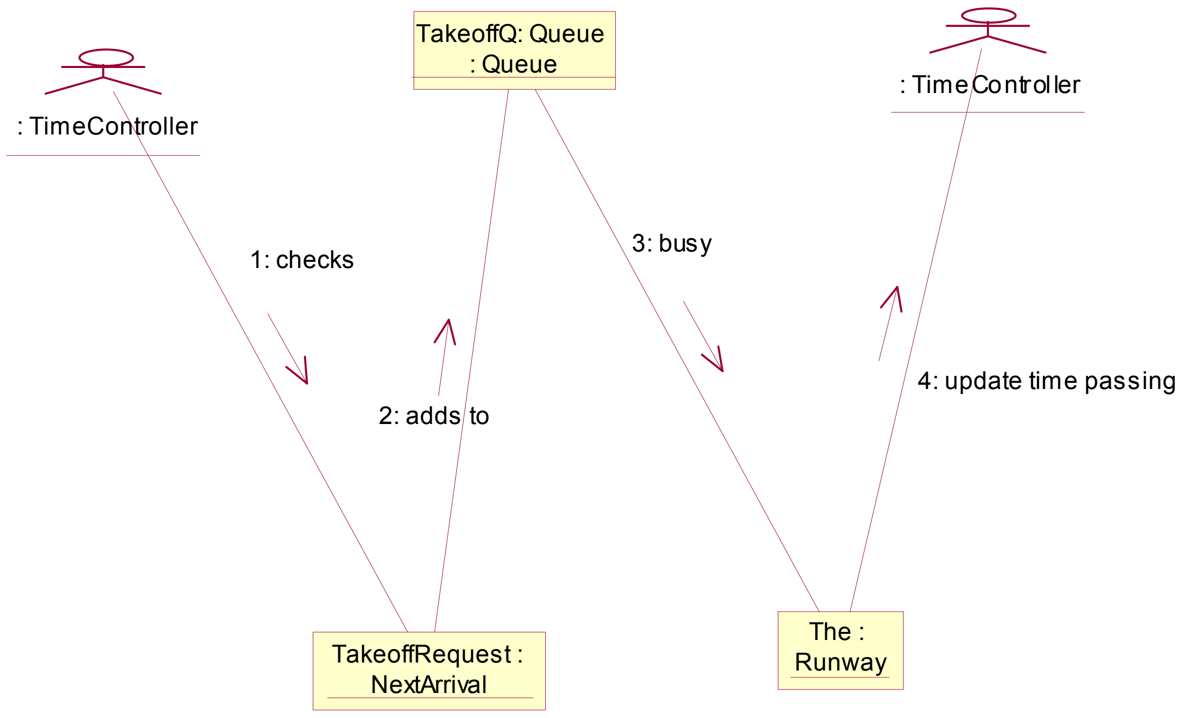


Figure 7 - Runway Busy Sequence

5.4.2 Runway Busy Collaboration (Figure 8)



<u>Figure 8- Runway Busy Collaboration</u>

5.5 Crash

5.5.1 Crash Sequence (Figure 9)

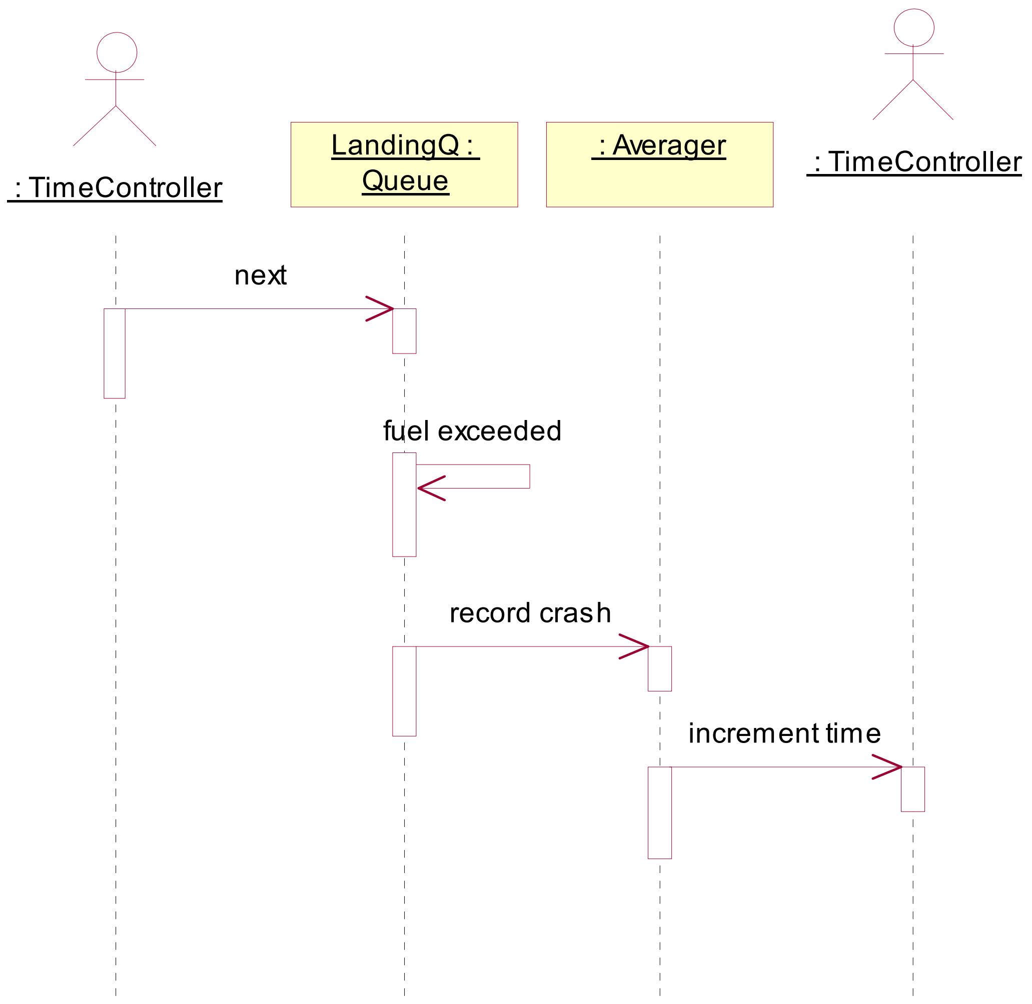


Figure 9 - Crash Sequence

5.5.2 Crash Collaboration (Figure 10)

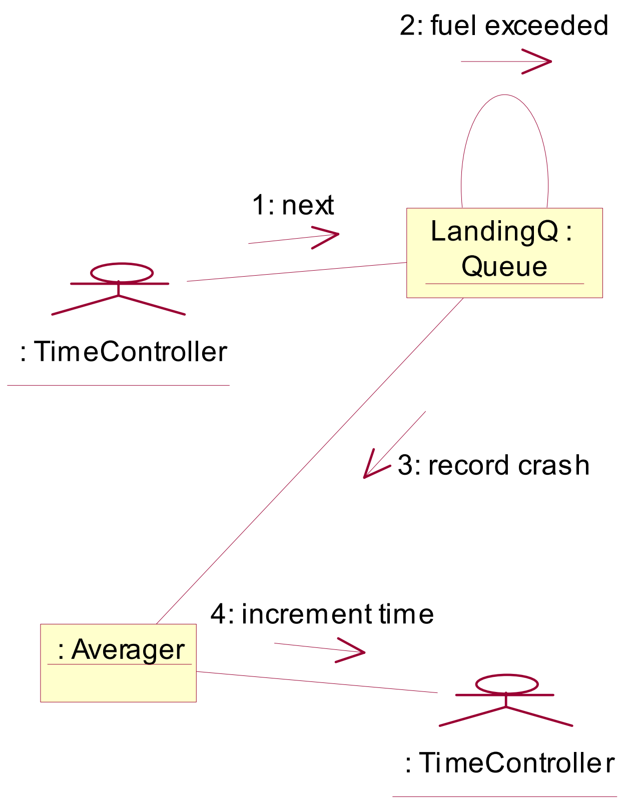


Figure 10 - Crash Collaboration

5.6 Print Report

5.6.1 Print Report Sequence (Figure 11)

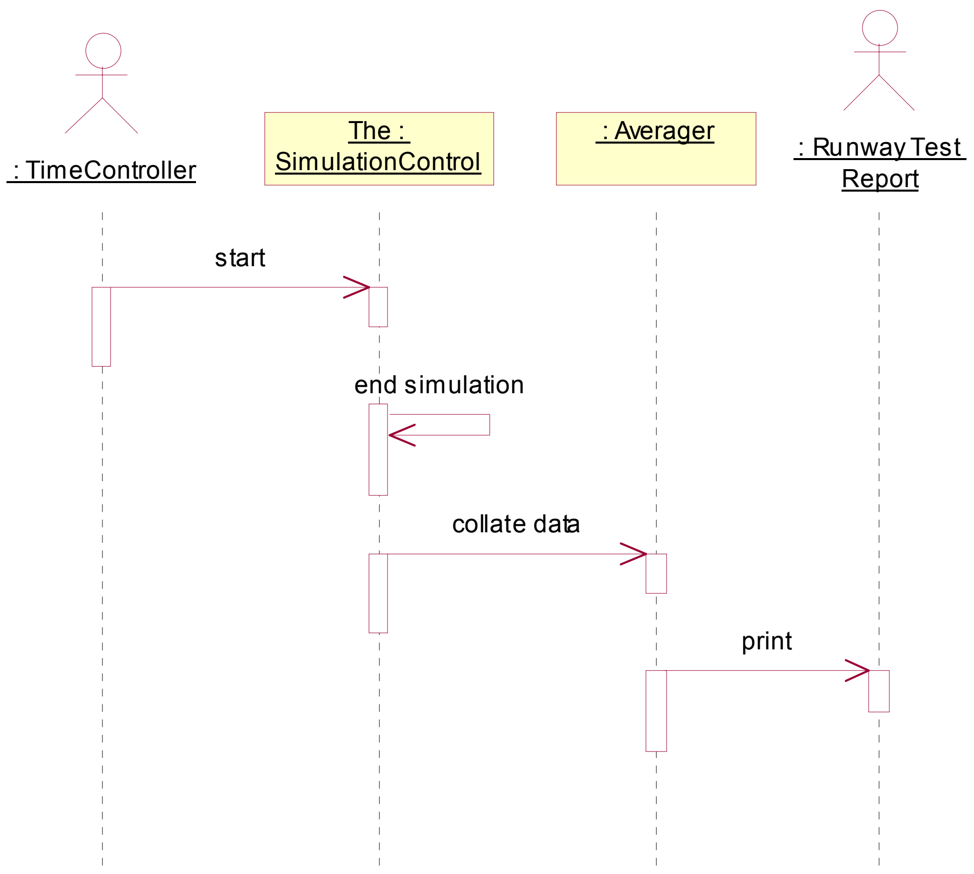


Figure 11 - Print Report Sequence

5.6.3 Print Report Collaboration (Figure 12)

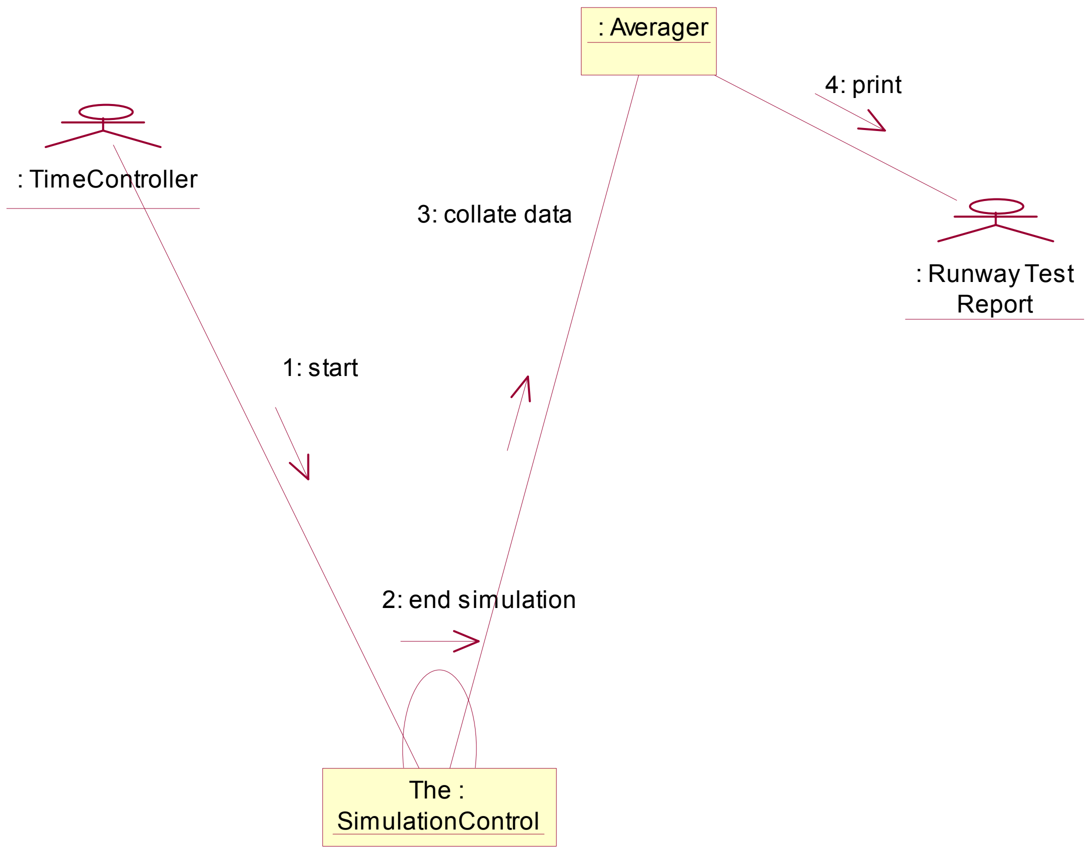


Figure 12 - Print Report Collaboration

6. REFERENCES

6.1 References

Appleton, Brad . A Software Design Specification Template. N.d. <http://www.enteract.com/~bradapp/docs/sdd.html>.

Booch, Grady, Ivar Jacobsen, and James Rumbaugh. The Unified Software Development Process (The Addison-Wesley Object Technology Series). 1st. Ed. New York: Addison Wesley, 1999..

GCC Home Page - GNU Project . Free Software Foundation. N.d. <http://gcc.gnu.org/>.

Sommerville, Ian. Software Engineering. 6th. Ed. New York: Addison Wesley, 2001.